This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method for semantic representation of one or more XML language inquiries across relational and non-relational data sources comprising:

receiving at least one inquiry;

defining a plurality of nodes of a graph structure which represents the at least one inquiry, the graph structure having at least one node object for every operation within the at least one received inquiry;

translating each of the at least one node objects using operators; and

generating a semantic representation having the graph structure;

wherein the semantic representation explicitly describes a meaning of the one or more XML language inquiries, the semantic representation including a tuple operator having three child nodes, the child nodes comprising a list of iterators that construct tuple space, a clause that filters the tuple space, and a clause that produces an outcome of the tuple space, wherein each iterator in the semantic representation of a tuple node corresponds to one column in the tuple space, and wherein the semantic representation decouples front-end language compilers from back-end query engines that use the semantic representation, such that, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations.

2. (Original) The method of claim 1, wherein the semantic representation is an intermediate language representation formed for interpretation and execution by a target query engine.

3. (Original) The method of claim 2, wherein the non-relational data sources comprise one or more of a text document, a spreadsheet, and a non-relational database.

4. (Original) The method of claim 1, wherein the generating step further comprises breaking down high level operations of the received inquiry into explicit parts.

5. (Original) The method of claim 4, wherein the explicit parts are common across multiple XML languages.

6. (Original) The method of claim 1, wherein the operators comprise one or more of special operators, data sources, literals, Boolean operators, sequence operators, arithmetic operators, string operators, value comparison operators, node comparison operators, tuple spaces, function definition and invocation, XML navigation, XML construction, XML property accessors, type operators, language specific operators, and data manipulation operators.

7. (Cancelled)

8. (Original) The method of claim 1, wherein the at least one received inquiry comprises one or more of an XML query language and an XML view definition language.

9. (Original) The method of claim 1, wherein the at least one received inquiry comprises one or more of an XPath, an XSLT, an XQuery, a DML, an OPath, and an Annotated Schema inquiry.

10. (Original) The method of claim 1, wherein the semantic language representation allows XML queries over XML views of relational data.

11. (Currently Amended) A semantics interpreter for expressing a meaning of one or more of an XML query and an XML view across multiple data sources comprising:

an input for receiving the one or more of an XML query and an XML view which form an inquiry;

a graph structure generator for defining node objects for every operation within the inquiry;

a translator for assigning operators for each node object wherein the operators break down operations of the inquiry into explicit parts; and

an output for providing the explicit parts as an intermediate language representation for expressing the meaning of the one or more of an XML query and an XML view, the intermediate language representation including a tuple operator having three child nodes, the child nodes comprising a list of iterators that construct tuple space, a clause that filters the tuple space, and a clause that produces an outcome of the tuple space, wherein each iterator in the ~~semantic~~ intermediate language representation of a tuple node corresponds to one column in the tuple space;

wherein the intermediate language representation decouples front-end language compilers from back-end query engines that use the intermediate language representation, such that utilization of the intermediate language representation in the semantics interpreter, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations.

12. (Original) The semantic interpreter of claim 11, wherein the multiple data sources comprise relational and non-relational data sources.

13. (Original) The semantic interpreter of claim 12, wherein the non-relational data sources comprise one or more of a text document, a spreadsheet, and a non-relational database.

14. (Original) The semantic interpreter of claim 11, wherein the operators comprise one or more of special operators, data sources, literals, Boolean operators, sequence operators, arithmetic operators, string operators, value comparison operators, node comparison operators, tuple spaces, function definition and invocation, XML navigation, XML

construction, XML property accessors, type operators, language specific operators, and data manipulation.

15. (Original) The semantic interpreter of claim 11, wherein the explicit parts are common across multiple XML languages.

16. (Original) The semantic interpreter of claim 11, wherein the intermediate language representation is formed for interpretation and execution by a target query engine.

17. (Currently Amended) A computer-readable <u>storage</u> medium having computer-executable instructions for performing a method of intermediate language representation of a received inquiry comprising:

receiving one or more of an XML query and an XML view forming the received inquiry;

defining a plurality of nodes objects in a graph structure which represents the at least one received inquiry, the graph structure having a node object for every operation within the received inquiry;

translating each node using operators which break down operations of the received inquiry into explicit parts; and

generating instructions corresponding to the explicit parts forming an intermediate language representation for subsequent queries over one or more of relational and non-relational data sources, wherein the intermediate language representation comprises an explicit description of a meaning of the received inquiry, and wherein the intermediate language representation decouples front-end language compilers from back-end query engines that use the intermediate language representation, such that utilization of the intermediate language representation, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations.

18. (Currently Amended) The computer-readable storage medium of claim 17, wherein the operators comprise one or more of special operators, data sources, literals, Boolean operators, sequence operators, arithmetic operators, string operators, value comparison operators, node comparison operators, tuple spaces, function definition and invocation, XML navigation, XML construction, XML property accessors, type operators, language specific operators, and data manipulation.

19. (Currently Amended) The computer-readable storage medium of claim 17, wherein the explicit parts are common across multiple XML languages.

20. (Currently Amended) The computer-readable storage medium of claim 17, wherein the received inquiry comprises one or more of an XML query language and an XML view definition language.

21. (Currently Amended) A computer system for generating a semantic representation of an inquiry comprising:

a processor for executing computer instructions and

at least one module comprising:

an input function for receiving one or more of an XML query and an XML view which forms the inquiry;

a graph structure generator for defining node objects for every operation within the inquiry;

a translator function for assigning operators for each node object wherein the operators break down operations of the inquiry into explicit parts; and

an output for providing the explicit parts as an intermediate language representation for expressing a meaning of the XML query and the XML view, wherein the intermediate language representation decouples front-end language compilers from back-end query engines that use the intermediate language representation, such that utilization of the intermediate language representation, when used in a compiler system having M front-front end languages and N back-end search engines, has a complexity of M plus N compiler implementations, and wherein the at least one module comprises one or more of one or more software modules and one or more hardware modules;

wherein the intermediate language representation includes a tuple operator having three child nodes, the child nodes comprising a list of iterators that construct tuple space, a clause that filters the tuple space, and a clause that produces an outcome of the tuple space, wherein each iterator in the ~~semantic~~ intermediate language representation of a tuple node corresponds to one column in the tuple space; and

wherein the intermediate language representation is executed directly by one execution engine, and is translated to SQL before execution by a second execution engine, and is executed partially in a third execution engine wherein a balance of the intermediate language representation is executed in a fourth execution engine.

22. (Original) The computer system of claim 21, wherein the operators comprise one or more of special operators, data sources, literals, Boolean operators, sequence operators, arithmetic operators, string operators, value comparison operators, node comparison operators, tuple spaces, function definition and invocation, XML navigation, XML construction, XML property accessors, type operators, language specific operators, and data manipulation.

23. (Original) The computer system of claim 21, wherein the explicit parts are common across multiple XML languages.